Antony Stang, Fiona Eberhart

Antony.Stang@students.fhnw.ch

Abstract

We present a lightweight, real-time clap detection system running on a Raspberry Pi. A simple 1D convolutional neural network classifies audio data from a USB microphone, achieving somewhat robust performance despite constrained computational resources.

Klatscherkennung

Dokumentation für den Projektauftrag von glaL4

Contents

[I. Einleitung 2](#_Toc194053918)

[Motivation und Kontext 2](#_Toc194053919)

[Zielsetzung und Vorgehen 2](#_Toc194053920)

[II. Projektaufbau und Hardware-Setup 2](#_Toc194053921)

[Hardware-Beschreibung 2](#_Toc194053922)

[Software-Umgebung 2](#_Toc194053923)

[III. Datenerfassung und Vorverarbeitung 2](#_Toc194053924)

[Datensammlung 2](#_Toc194053925)

[Vorverarbeitungsschritte 2](#_Toc194053926)

[IV. Modellentwicklung 2](#_Toc194053927)

[Architekturüberblick 2](#_Toc194053928)

[Implementierungsdetails 2](#_Toc194053929)

[Training 3](#_Toc194053930)

[V. Ergebnisse und Auswertung 3](#_Toc194053931)

[Trainingsergebnisse 3](#_Toc194053932)

[Fehlerraten und Beispielanalysen 3](#_Toc194053933)

[Echtzeitverhalten 3](#_Toc194053934)

[VI. Diskussion 3](#_Toc194053935)

[Interpretation der Resultate 3](#_Toc194053936)

[Limitationen und mögliche Verbesserungen 3](#_Toc194053937)

[VII. Fazit und Ausblick 3](#_Toc194053938)

[Zusammenfassung 3](#_Toc194053939)

[Zukünftige Entwickelung 3](#_Toc194053940)

# Einleitung

## Motivation und Kontext

-Beschreibung des Zieles, hier: Ich will mein PC mit Klatschen anschalten können. Dafür wird ein Raspi eingesetzt, welcher ein Wake on Lan Signal an mein PC sendet, sobald er Klatschen erkennt.

## Zielsetzung und Vorgehen

-Ziel ist es im Rahmen des glaL4 Projektes ein Neurales Netz ausreichend zu trainieren um das Klatschen zu erkennen.

-Zusätzlich sollen verschiedene Herandgehensweise getestet werden, um das effizienteste zu identifizieren

Es werden 3 verschiedene Verfahren ausgestet, welche hier verglichen werden

* Ein Neurales Netz welches einen Zeitabschnitt als Audiosignal erhält ohne Vorverarbeitung der Daten
* Ein Neurales Netz welches auf 32 Merkmale (max dB, min dB, Flankensteilheit sind aus dem Zeitsignal gefiltert) trainiert wird
* Ein Neurales Netz welches einen Zeitabschnitt als Fourier transformiertes Signal erhält

Jedes dieser Neuronalen Netze wird in einem Versuch experimentell so eingestellt, bis der Training Loss zufriedenstellend klein sind. Eingestellt werden Anzahl Neuronen, Anzahl Hidden Layers und maximale Anzahl Epochen.

# Projektaufbau und Hardware-Setup

## Hardware-Beschreibung

-Raspi 5, USB-Mikrofon welches ich in einer Schublade gefunden habe, Lan Kabel um Wake on Lan ohne auseinandersetzten mit dem Heimnetz umzusetzen.

## Software-Umgebung

-Raspi OS, die ganzen Bibliotheken der Python3 Umgebung, welche ich eingebunden habe.

# Datenerfassung und Vorverarbeitung

## Datensammlung

-Audioaufnahmen von je 60sec welche manuell zugeschnitten wurden um Klatscher markiert rauszuschneiden.

-Hintergrundgeräusche wurden über Laute Musikvideos und Normallauten Serien erzeugt.

## Vorverarbeitungsschritte

-Trimmen/Padding der Audiosignale auf 4410 Daten (100ms länge bei 44,1kHz Mic)

-Normalisierung in Form von Stereo auf Mono

-FFT wurde bei einem Versuch angewandt um das Getrimmte/paddete und Normalisierte Signal weiterzuverarbeiten

# Modellentwicklung - Fiona

Architekturüberblick

-Versuch 1: 2 Conv-layer (mit 8800 Übergabemerkmale / 1 Hidden Layer im FC. (genaue daten aus dem Code auslesen pls)

-Versuch 2: 1 kleine Conv-layer (mit nur 32 Übergabemerkmale) /2 Hidden Layer im FC. (ebenfalls genaue daten noch raussuchen)

-Versuch 3: selbes netz wie Versuch 1 (hier halt nur mit den FFT daten weshalb der erste Konvolution Layer einen anderen Input hat

Implementierungsdetails

-Verwendetes Framework (Pytorch denke ich)

-Hyperparameter (Learning Rate, Epochen, Abbruchskriterien)

Training

-Train-/Validation Split (bei uns 80/20), Loss-Funktion, Optimizer hat mir GPT vorgeschlagen, weis nicht was das genau in unserem Projekt macht.

-Training war auf dem Pi / Zeit pro trainingsschritt war nicht nennenswert (im Bereich 1 – 3 Sekunden) – deshalb war es nicht nötig das netz extern auf der GPU zu trainieren.

# Ergebnisse und Auswertung - Fiona

Trainingsergebnisse

-Ich mache noch Screenshots von den typischen Trainingsverläufen.

-Vergleiche zwischen den 3 Trainingsverläufen

Fehlerraten und Beispielanalysen

-Am besten den Fehler irgendwie Visualisieren

Echtzeitverhalten

-Vermerk das noch keine Langzeitanwendung umgesetzt wurde. Bei kurzen Tests wurden bereits eine nervige Menke False Positives Festgestellt.

# Diskussion

## Interpretation der Resultate

-Selbsterklärend XD

## Limitationen und mögliche Verbesserungen

- Limitation ganz klar zu wenige Daten

- Der Datensatz ist ebenfalls noch nicht ausreichend dynamisch – es fehlen daten von Ausnahmefällen wie Bohren, Silvester, Lautes Streiten, den kleinen Zeh am Tisch anschlagen usw.

# Fazit und Ausblick

## Zusammenfassung

-Haben wir unser Ziel erreicht? (Jain – es fehlen mehr daten für ein robustes System jedoch ist der Ansatz vielversprechend)

-Welche Erkenntnisse ziehen wir aus dem Projekt? (weniger Übergabeparameter vom Konvolution Layer können bis zu einem gewissen punkt gut mit mehr und komplexeren FC Layer kompensiert werden) (fft hat einen geringen Vorteil beim Trainieren des Netztes gegeben, in form von weniger Cyclen, hat aber nach dem Training kein Vorteil mehr für die Anwendung)

## Zukünftige Entwickelung

-Weitere Arbeit an dem Projekt (kurz erwähnen welche IRL-Anwendung ich hierfür habe).